INTERNATIONAL DESIGN CONFERENCE - DESIGN 2018
https://doi.org/10.21278/idc.2018.0236

AGILITY FACTORS AND THEIR IMPACT ON PRODUCT DEVELOPMENT PERFORMANCE

E. Rebentisch, E. C. Conforto, G. Schuh, M. Riesener, J. Kantelberg, D. C. Amaral and S. Januszek

Abstract

Agile product development is popular but still not well understood beyond its methodological implications. This paper reports an identification of agility factors enabling teams to make and communicate decisions more quickly in product development. On this basis, a quantitative System Dynamics (SD) model is developed and analyzed to explore the impact of the identified agility factors on the product development performance. The results show that both methodological and team factors do not only influence agility but can also significantly improve the project completion time and product quality.

Keywords: product development, agile development, teamwork, system dynamics, collaborative design

1. Introduction

Highly dynamic market opportunities and rising global competition have led to a shift from seller's to buyer's market. As a consequence, today's companies and organizations need to be more flexible and reactive towards changes and perceptive towards new market opportunities to remain competitive and innovative (Schuh and Bender, 2012; Schwab et al., 2016). Product development organizations are also affected, as they are consistently exposed to a high degree of uncertainty in terms of changing customer requirements or other instable market conditions (Hull et al., 2017). That is why companies and their product development teams need to move from an anticipatory to adaptive style of developing new products, especially in a highly technological and innovative business world (Highsmith, 2010; MacCormack, 2013; Schuh et al., 2017).

Against this backdrop, agile product development is considered to be an appropriate management approach for product development which has been applied successfully across different industries (Conforto et al., 2014a ; Schuh et al., 2016). Its methodological application including different practices and tools is well understood, lesser is its promise of improving project performance by promoting agility. Conforto et al. (2016) made a first step towards a better understanding by defining the "agility" construct as an ability of the project team to quickly change the project plan. Nonetheless, there still remains a lack of understanding how agile methods or other factors can lead to more agility and how this might in turn improve the project performance.

This paper therefore attempts to better understand agility in product development projects by identifying agility factors that help teams to make fast decisions and communicate them quickly as a response to changing project conditions. Additionally, this research tries to quantitatively assess the impact these factors have on the time and quality performance of product development projects.

First, this paper investigates literature (Section 2) and empirical data (Section 3) to understand how methodological and team factors of a product development project improve a team's ability to react to project changes. Based on this knowledge, a System Dynamics (SD) model is developed to theoretically replicate the dynamic behavior of a product development system considering the interdependencies between different system variables with the help of feedback loops and mathematical equations (Section 4). The model is then used to simulate different project scenarios and to analyze the impact that various project characteristics have on the project performance (Section 5). The article concludes with an outlook on further research opportunities (Section 6).
The literature review and empirical data analysis indicated that both the product development method and team characteristics influence the agility of a product development team. Furthermore, the analysis of the SD model simulations showed strong impacts of methodological factors on time performance. In turn, team factors showed a significant effect on the quality of the developed product.

2. Agility in product development: Fundamentals and theories

The product developing organization, which is responsible for providing the product to the market, is generally surrounded by numerous stakeholders including the customer whose desires and expectations ultimately need to be satisfied (Ulrich and Eppinger, 2015). However, meeting the customer requirements has become a very challenging endeavour in today's turbulent market situation. Product development is usually carried out as a project which has to be well managed, not exceed a previously defined budget, and be run within a limited time frame. During this time frame, unexpected project changes might appear and challenge the product development team. Being responsive to these changes can be crucial for the product development project's success (Highsmith, 2010).

2.1. Defining "agility" in the product development context

Conforto et al. (2016) conducted a comprehensive literature review and frame semantic analysis to find a complete definition of the "agility" construct. The results defined agility as "a project team's ability to quickly change the project plan as a response to customer or stakeholder needs, market or technology demands in order to achieve better project or product performance in an innovative and dynamic project environment" (Conforto et al., 2016, p. 667). Transferred to the world of product development, agility thus can be described as the capability to discover and understand changing product requirements, and being able to quickly consider these changes while making progress in developing the product. This capability requires a good level of perception towards the project's stakeholders and on the other side high decision-making speed when the team discovers potential product changes and needs to decide whether they should be implemented or not.
In other words, the agility of a product development team describes how well it is dealing with uncertainties in terms of potential product changes which nowadays have become a latent risk in many product development projects (Hull et al., 2017). From the eyes of a product development team, uncertainties are either a lack of knowledge or a lack of definition (McManus and Hastings, 2006). In case of a lack of knowledge, the team is not aware of information that is needed to complete the project in a rational way. In case of a lack of definition, certain information about the product is still in question and needs to be clarified, specified, or decided in order to complete the project. According to the categorization of knowledge by Donald Rumsfeld, these two classes of uncertainty will be defined as *unknown unknowns* and *known unknowns* (Asan and Bilgen, 2013). *Unknown unknowns* are thus treated as project tasks that are unexpected and surprising to the project team. In contrast, *known unknowns* are tasks the project management is aware of and which it therefore can consider while planning and running the project (Ramasesh and Browning, 2014). Accordingly, agile product development goes beyond the sole application of methods and is about uncovering and processing *unknown unknowns* and managing *known unknowns* in a way so that the project can be completed on time.

2.2. Product development methods: Waterfall vs. agile development

The application of structured process models to product development facilitates the methodological study and improvement of development processes. Additionally, it helps to match the right management

approach with the project's environment which is important for a project to be successful. Therefore, many different process models have been developed and adapted to different business environments over the past decades. Two of the most popular approaches will be introduced in the following and discussed with regard to their leverage on agility.

Traditionally, most companies used to develop their products by means of deterministic-normative approaches, following a sequential process structure. Manufacturing companies used the phase-gate approach, where the development of the product moves from one phase to the other by fulfilling quality criteria and passing the respective quality gate (Cooper, 2008). Software industry on the other hand used similar approaches, for example the Waterfall model (see Figure1) (Royce, 1970). All of these sequential methods have in common that they dedicate much effort to the planning phase of a development project to identify all requirements and specifications of the product so that the project can be run strictly according to the defined project phases (Vinekar et al., 2006).

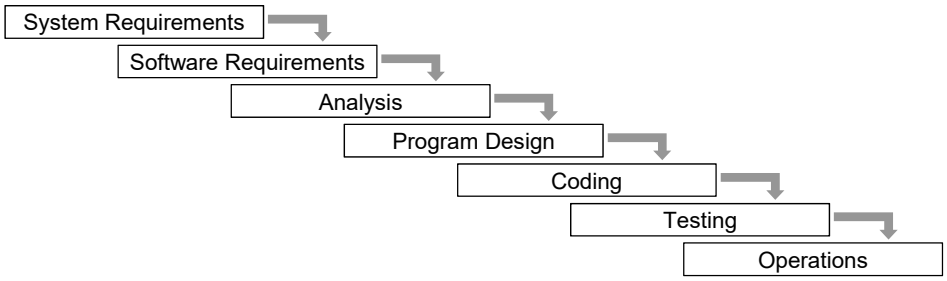

Figure 1. Waterfall model by Royce (1970)

The project management approach of agile development has arisen since 2001, when the Agile Manifesto for Software Development, a collection of so called "agile" or lightweight methods, was defined by a group of practitioners from the I.T. sector (Beck et al., 2001). The roots of agile development go back even further and can be traced back to the development of methods like *Scrum* or *Extreme Programming (XP)*. These methods comprise various practices which have their own benefits and can be combined with each other. As they all incorporate the ideas of the Agile Manifesto, they are called "agile methods" and share fundamental commonalities: They all develop products incrementally in short iterations, deliver regular product releases (deployable solutions) to their stakeholders, and emphasize small and co-located teams (Smith, 2007). Figure 2 exemplarily depicts the Scrum process and visualizes its highly iterative character with two circles. The bigger circle represents so called sprints, short periods of time in which small parts of the product (increments) are developed and after being reviewed ready to be released to the customer. The smaller circle represents daily meetings to update the team and discuss the project progress or current issues (Cohn, 2010).

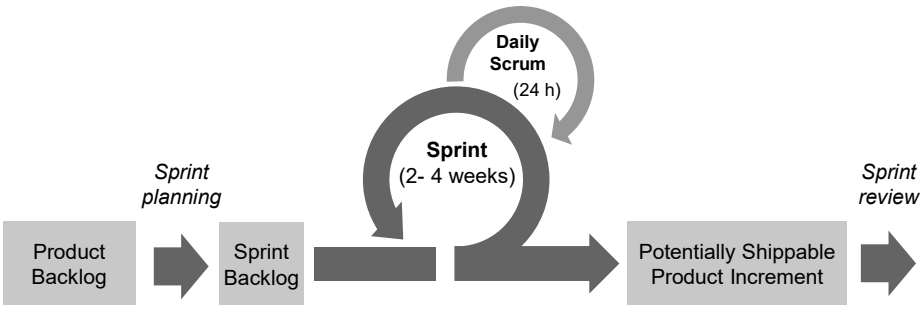

Figure 2. Scrum process

Comparing the two main approaches, major differences with respect to their leverage on agility can be identified. Sequential process models like the Waterfall model are criticized for their rigidity resulting from the high level of detail in their planning phase (Serrador and Pinto, 2015). Due to high process formality, they lack flexibility and require much effort and time to react to changes (Kock and Gemünden, 2016). Furthermore, these models do not involve the customer constantly in the

development process, which can reduce the awareness and delay the discovery of potential exogenously-defined changes (Turner, 2007; Asan and Bilgen, 2013). By contrast, the agile development approach involves the customer regularly and proactively into its process and asks for feedback which is then considered in further development steps. Thanks to its highly iterative style of development and openness towards changes, it is considered to be more reactive and therefore well-suited for companies which are exposed to high risks of product changes (Conforto and Amaral, 2016).

2.3. Agility enabling team characteristics

As agility is considered a team's capability, certain team characteristics can be identified as leverage points on agility. In this context, team adaption theory is of high relevance, as it describes how well a team is able to adjust its strategies based on information gathered from its environment (Salas et al., 2005). Particularly for product development teams who face poorly specified product requirements and are exposed to a high risk of potential changes, their adaptability is an essential prerequisite for high performance (Schmidt, 2016). Following the teamwork theory by Salas et al., which has been proved to be applicable to (agile) product development as well (Strode, 2015; Schmidt, 2016), *team orientation*, *team leadership*, *mutual performance monitoring*, and *backup behavior* are important factors for developing team adaptability (Salas et al., 2005; Schmidt, 2016).

Accordingly, the team size has a significant effect on agility because it promotes better communication and coordination. Small teams encourage direct one-on-one communication, improve the alignment between team members, facilitate conflict management, and enhance the sense of commitment and responsibility (Trott, 2005). A higher awareness of each other's work and a high level of collaboration contribute to mutual performance monitoring and backup behavior (Schmidt, 2016). Furthermore, smaller teams can more easily make timely, informed decisions (Trott, 2005), which is critical in a dynamic environment and supportive for developing agility.

Team empowerment, *team autonomy*, and *self-organization* are terms being used synonymously and describe the team's authority and responsibility to plan, schedule, and assign tasks to team members and to make decisions related to the project and especially the product (Moe et al., 2008). A self-managed team does not have a project manager who makes or approves all important decisions. Rather, the team itself has enough decision-making authority and can thus influence team effectiveness directly. This can in turn increase the speed and accuracy of solving operational problems and reduce uncertainty (Moe et al., 2008; Hoda et al., 2012). Team autonomy is therefore considered to also have a positive effect on agility.

3. Finding empirical evidence for agility factors

To find empirical evidence for the factors enabling agility (agility factors), research results from the "Project Management Agility Global Survey" were provided by Conforto et al. and analyzed (Conforto et al., 2014a; Conforto et al., 2014b). The study comprised various questions supposed to identify the surveyed companies' use of agile management practices in product development projects and their performance. Out of all questions, 45 questions were considered as potentially relevant for promoting agility and selected to conduct a Spearman's correlation analysis. Two of the questions referred to the *time needed to analyze changes and make a decision* and the *time needed to update the project plan and communicate changes*. Since fast decision-making and quick communication are prerequisites of agility (Schmidt, 2016), these two variables were considered as agility measurement metrics. Variables that were represented by the remaining questions and showed a significant correlation, either at the 0.05 level (2-tailed) or at the 0.01 level (2-tailed), were chosen for a multiple linear regression analysis. After conducting a first regression analysis, variables with a high p-value were sorted out and the remaining variables were kept and defined as agility factors. The following table (Table 1) shows the independent variables (agility factors) and the two dependent variables (agility measurement metrics) including the corresponding results from a final regression analysis. Each variable was named according to the context of its corresponding question in the survey. Moreover, the analysis was done in a converted scale which explains why all coefficients are positive. Therefore, the definition of the variables and their scales will be explained further below.

Table 1. Overview of regression analysis results – agility factors (Conforto et al., 2014a)

		unstandardized coefficients		standardized coefficients		
dependent variable	independent variable	B	Std. Error	Beta	t	Sig.
time to analyze changes and make a decision (TACMD)	(constant)	2.248	.278		8.079	.000
	team size	.100	.024	.139	4.092	.000
	team autonomy	.092	.030	.107	3.120	.002
	executive support	.109	.036	.104	3.056	.002
	fraction of initially known unknowns	.092	.028	.113	3.331	.001
	release cycle time	.116	.027	.146	4.354	.000
time to update the project plan and communicate changes (TUPPCC)	(constant)	1.387	.319		4.347	.000
	team size	.131	.024	.183	5.521	.000
	team autonomy	.131	.028	.153	4.616	.000
	executive support	.093	.035	.089	2.695	.007
	team meeting frequency	.218	.041	.174	5.279	.000
	release cycle time	.120	.026	.151	4.576	.000

TACMD: The first dependent variable describes how much time was required by the project team to meet, analyze and make a decision after discovering a potential product or project change. The regression analysis was done in a dimensionless scale starting from *1* to *7*. To rescale its values back into the surveyed scale, the regression scale from *1* to *7* was rescaled asymptotically to a graduated scale starting from *30 or more days* to *less than 24 hours*.
TUPPCC: The second dependent variable describes how much time was required by the project team to rework the project plan and communicate the changes to the team, managers, and stakeholders. According to the first dependent variable, its values were also rescaled asymptotically from a scale from *1* to *7* to the graduated survey scale starting from *30 or more days* to *less than 24 hours*.
Team size: The team size describes how many members were involved in one team. Its values were scaled linearly from a graduated scale starting from *more than 36 people* to *less than 6 people* to a dimensionless scale from *1* to *7*.
Team autonomy: The team autonomy describes how often the project team members were allowed to define and execute a new plan when facing changes. Its values were scaled linearly from a graduated scale starting from *0% of changes* to *100% of changes* to a dimensionless scale from *1* to *7*.
Executive support: The executive support describes the executive's or management's commitment to support the project execution. Its values were scaled from a graduated scale starting from *limited commitment* to *full commitment* to a dimensionless scale from *1* to *7*.
Fraction of initially known unknowns: The fraction of initially known unknowns describes the level of detail of the initial project's scope description, representing the amount of unspecified product requirements. Its values were scaled from a graduated scale starting from *detailed description (changes seemed to be negative and to be avoided)* to *brief description (welcoming changes)* to a dimensionless scale from *1* to *7*.
Team meeting frequency: The team meeting frequency describes how often the project team was meeting to discuss project related topics (e.g. progress, issues, ideas.). Its values were scaled asymptotically from a graduated scale starting from *less than once every two months* to *every day* to a dimensionless scale from *1* to *7*.
Release cycle time: The release cycle time describes the delivering frequency of project's partial results to stakeholders. Its values were scaled asymptotically from a graduated scale starting from *more than 6 months* to *weekly* to a new, dimensionless scale from *1* to *7*.

According to these results, an agile team is characterized by a small team size, high team autonomy, frequent team meetings, and a high commitment by the organization's executives for developing the product and is therefore able to make quick decisions and to communicate them quickly as well. Furthermore, a brief project description which encourages the team to implement changes plus frequent release cycles which are used to get feedback from stakeholders further promote the fast and successful consideration of product changes and are therefore promoting the team's agility.

4. Developing a new SD model

System Dynamics (SD) is a modelling method which allows to design, model, simulate, and analyze dynamic and complex systems with respect to their process, information, organizational boundaries, and strategies. Out of many different modelling methods, SD was considered a good choice because it allowed to represent multiple work constraints, iterative flows of work and different performance dimensions. It is based on the assumption that the behavior of a system is largely caused by its structure, whereby a system consists of multiple elements and their interactions (Sage and Rouse, 2009).

4.1. The main subsystem of the new SD model

The central subsystem of the developed model is based on the classic rework cycle, which was developed by Cooper and has since then become the impetus of a wide study of project management in SD (Cooper, 1993). The classic rework cycle consists of stock and flow variables which describe a project or a single project phase. Stock variables (e.g. *Work to be done*) describe accumulations of work, and the flows (e.g. *work being done*) enable the transformation of work (see Figure 3). Project tasks which are completed incorrectly, move into the *undiscovered rework* stock and need to be discovered first before being fed back into the cycle. A project is completed when enough tasks have been processed from the initial stock *Work to be done* into the *Work really done* stock (Cooper, 1993).

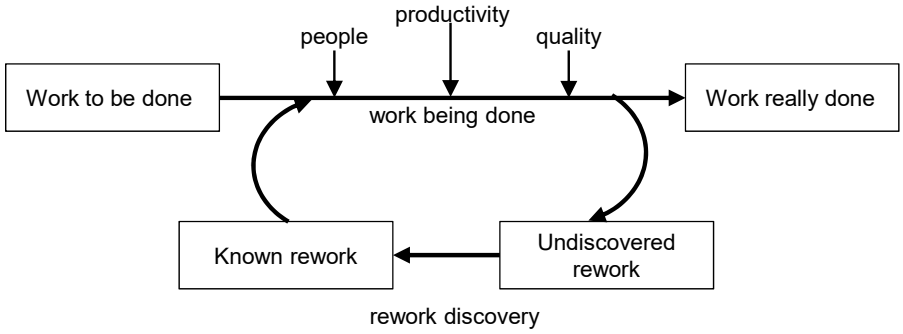

Figure 3. The classic rework cycle

Instead of one single stock variable describing the project scope, three different types of backlogs were built into the rework cycle. The *product backlog* represents a list of product requirements (measured as tasks). Depending on the size of the single product releases, during each release cycle, a fixed number of these requirements is processed to the next stock variable, the *release backlog*. Depending on the number of iterations per release cycle, the release backlog is split again into single *iteration backlogs* that are completed one after another (Glaiel, 2012). By processing all tasks from the *product backlog* into the *iteration backlog* at once and completing the project within one iteration cycle, product development can be simulated according to a sequential process model. In contrast, a highly frequent style of product development, resembling the agile development approach, can be simulated by separating the *product backlog* into multiple *release backlogs* and even smaller *iteration backlogs* which would then be completed separately in the course of one project (Glaiel, 2012).

In addition, the agile rework cycle was further extended by two more kinds of project scope to replicate the detection of changes and reduction of uncertainties which represents a major challenge of product development projects. Apart from the stock variable *product backlog* (*known knowns*), *known unknowns* and *unknown unknowns* were modelled as further stock variables according to their definitions in Section 2 (see Figure 4). Sometimes, parts of the project scope are not known precisely or not known at

all at the very beginning of a project. If this information can be expected and controlled by the team, it is categorized as *known unknowns*. An example for *known unknowns* is the awareness of a certain product requirement (e.g. size, weight) which is not specified yet and impedes further work or the specification of other requirements. Decisive for the model is that this information is mainly discovered through internal work and thus driven by iterations, and has an impact on the productivity of the team (Asan and Bilgen, 2013). In contrast, *unknown unknowns* are information that is neither known nor can be expected by the team. The discovery of *unknown unknowns* is predominantly realized through the feedback from external stakeholders (driven by releases), but also through internal discussions or product tests (driven by iterations) and through error discovery (Li, 2012). The model assumes that the discovery of *unknown unknowns* leads to changes in the product (Sterman, 2003).

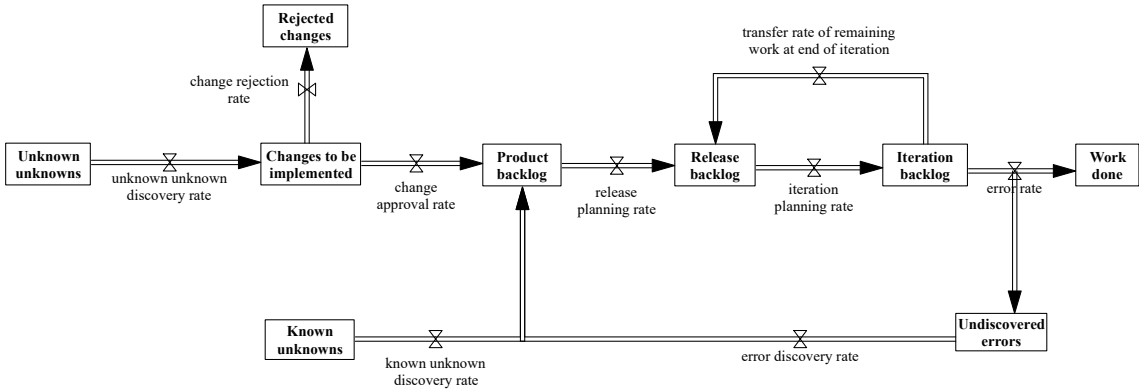

Figure 4. Main subsystem of the SD model

To make use of the quantitative data gained from the empirical data (Section 3), the *change approval rate*, which is responsible for processing changes in the model, is generally defined as follows:

$$change\ approval\ rate = \frac{Changes\ to\ be\ implemented * change\ acceptance}{TACMD+TUPPCC} \quad (1)$$

$$TACMD = 2.248 + 0.1 * team\ size + 0.092 * team\ autonomy + 0.109 * executive\ support + 0.092 * fraction\ of\ initially\ known\ unknowns + 0.116 * release\ cycle\ time \quad (2)$$

$$TUPPCC = 1.387 + 0.131 * team\ size + 0.131 * team\ autonomy + 0.093 * executive\ support + 0.218 * team\ meeting\ frequency + 0.12 * release\ cycle\ time \quad (3)$$

The *change acceptance* is a normalized variable which determines the relative amount of changes that is not rejected, but accepted and further processed to the *product backlog*.

To complete the model and make it functional, even more variables and subsystems were defined. The mechanisms of the rework cycle, the discovery and management of changes, and the behavior of the team were for example driven by variables which were grouped as project characteristics. Despite their importance for the model's behavior, these other variables and subsystems will not be introduced in detail within this paper because a full discussion would go beyond the length limitations.

4.2. Description of the model physics

As the model describes the dynamics of a product development project, the key elements of the model are time, workforce, and the backlog. The backlog represents the project scope measured in tasks and the workforce represents the whole project team measured in people. Time is measured in weeks. The model's behavior is mainly driven by the agile rework cycle which processes tasks with the speed of the workforce's productivity. Most model variables change their value over time. The backlog either increases with new tasks being added to the project, for instance by discovering errors that cause rework or changes that have to be implemented, or decreases by the completion of tasks. The overall remaining tasks are stored in the product backlog. Depending on the iteration and release cycle times, more or less tasks flow from the product backlog to the release or iteration backlog at the beginning of each cycle (Cooper 1993; Glaiel, 2012).

To finish a project, the project progress needs to reach the threshold of 95% which means that all known tasks (*known knowns*, *known unknowns*) have to be completed correctly. Additionally, all *unknown unknowns* must be smaller than or equal 10% of their initial value. These conditions guarantee that most of the development tasks have been completed and almost all relevant development tasks, that the project team was not aware of before starting the project, will be discovered and either accepted and added to the backlog or rejected.

5. Simulation results and discussion

Testing the model and making on-going improvements to better understand the variables, interactions, and to debug the model is an iterative process that runs simultaneously to the modelling process. This SD model was examined by conducting various tests (i.a. parameter assessment, extreme conditions, structure assessment, boundary adequacy, dimensional consistency, sensitivity analysis, etc.). However, the tests showed positive results and gave confidence for the model's validity.

5.1. Introducing the analysis framework

The rationale of the simulations was to understand when different development approaches best apply and when they do not. Therefore, a scenario analysis was conducted to compare the performance outcomes of different project scenarios which were represented by specific sets of model parameters. Initially, four main scenarios with the same number of total tasks (500) were defined according to different uncertainty levels. Two extreme scenarios were chosen to represent a very certain project scenario with no changes (scenario I) and an extremely uncertain project where the team has to face a very high number of changes (scenario IV). The two other scenarios II and III were chosen to determine how the two development approaches deal with more moderate levels of uncertainty.

Table 2. Overview of the four main project scenarios

	Scenario I	Scenario II	Scenario III	Scenario IV
Known knowns	500 tasks	300 tasks	100 tasks	50 tasks
Known unknowns	0 tasks	100 tasks	100 tasks	0 tasks
Unknown unknowns	0 tasks	100 tasks	300 tasks	450 tasks

Additionally, further sub-scenarios were defined by two sets of further variables, describing the product development approach and the team characteristics, with two different configurations each, resulting in four sub-scenarios (see Table 3).

Table 3. Overview of the four sub-scenarios

	agile team, short cycles	non-agile team, short cycles	agile team, long cycles	non-agile team, long cycles
iteration cycle time [measured in weeks]	1	1	20	20
release cycle time [measured in weeks]	1	1	20	20
team size [measured in persons]	3.33	10	3.33	10
team autonomy [0 = no decision-making authority; 1 = full decision-making authority]	1	0	1	0
executive support [0 = limited commitment; 1 = full commitment]	1	0	1	0
team meeting frequency [measured in meetings/week]	5	0.5	5	0.5

First, the iteration and release cycle time constitute either a long-cycled product development approach with cycle times of 20 weeks or a highly iterative approach with cycle times of one week. The second set of variables refers to team characteristics and constitutes two team configurations with regard to their decision-making and communication speed. The relevant team characteristics were selected according to the results from the regression analysis and represent either an agile team, being able to make decisions quickly and communicate them quickly, or a non-agile team with weaker decision-making and communication skills. Notably, the team size describes the average number of team members in one team, whereby the overall project can be run by multiple teams. Thus, the overall project staff size remained constant within all scenarios, solely the number of teams changed according to the changing team size.

5.2. Time performance

Figure 5 gives an overview of the scenario analysis results on the project duration for every sub-scenario. Every four clustered bars represent the results of one sub-scenario. The numbers above the bars indicate the respective main scenario. Increasing Roman numerals indicate a higher risk of product changes, per the scenarios described in Table 2. Agile teams are represented by light grey bars and non-agile teams are darker-colored. For instance, the first bar on the left side of the chart represents the time needed by an agile team that works in short iteration cycles on a very certain project to complete the project.

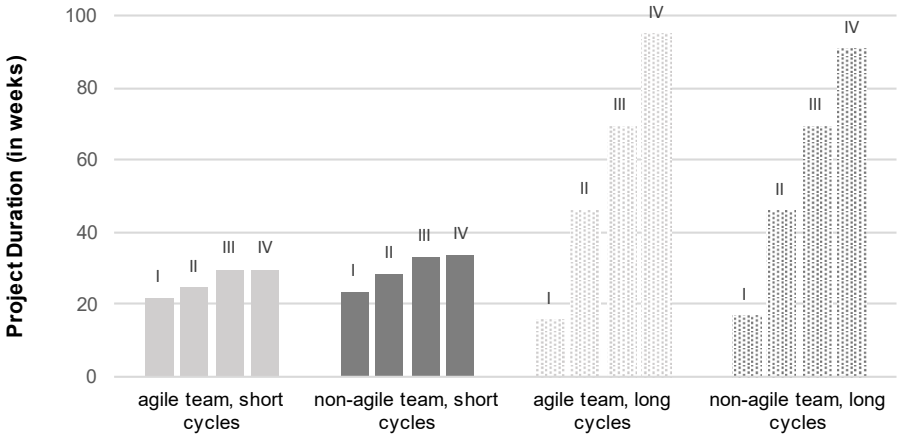

Figure 5. Scenario analysis results for time performance

The smaller the bar is, the lesser time was needed to complete the project, which in turn means better project performance. In the first project scenario (I), long iteration and release cycles led to better project performance. In case of the three more uncertain project scenarios (II, III; IV), short cycle times performed significantly better which confirms the theory. Team characteristics did not show a big impact on the time performance compared to the variation of cycle times. Therefore, the development method is considered to have more leverage on time performance than team characteristics.

In scenario I, the long-cycled development approach led to better time performance because of efficiency losses and more rework caused by highly frequent iterations and releases. When the certainty level is high and no product changes occur, an intense involvement of the customer in the project does not provide much benefit, but rather time losses. Moreover, the development of many prototypes and conduction of a high number of product tests causes much (rework) effort which is also not very beneficial in this case (Petersen and Wohlin, 2009). According to the model behavior, a well-planned project is completed about 25% faster when it can be run within one development cycle. However, in case of uncertainty and changing product requirements, the highly iterative approach reveals its advantages against long development cycles. The iterative approach steadily reduces uncertainty with the help of frequent product tests and active customer involvement which results in early discovery of changes and their implementation into the product (Asan and Bilgen, 2013). By contrast, the second approach shows issues with completing the project in a timely manner. The reason is that a long-cycled

approach requires too much time to detect changes, so that they occur in relatively late stages of the project and require much rework effort to be implemented (Smith 2007; Li, 2012).

5.3. Quality performance

The results of the second performance dimension are visualised in Figure 6 and describe how many errors remained undiscovered at the end of the project completion. To perform well in this performance dimension, either at least the *error rate* has to be low or the *error discovery rate* has to be high. This performance dimension is therefore considered a measure of the quality performance (Glaiel, 2012). Smaller bars again represent better project performance. Additionally, it is recommended to focus on the comparison of the performance numbers with each other, as the absolute numbers do not have a significant meaning for the analysis.

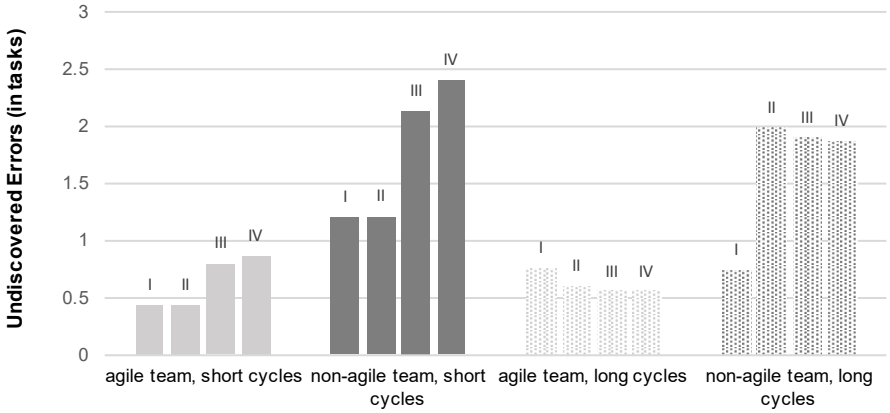

Figure 6. Scenario analysis results for quality performance

The iterative approach, characterized by short development cycles, indicates that the quality performance decreases with growing uncertainty. For both its sub-scenarios (*agile team, short cycles* and *non-agile team, short cycles*), the last two project scenarios end with significantly more undiscovered errors than the first two project scenarios. On the contrary, a long-cycled development approach shows a different behavior. More uncertainty leads to slightly fewer undiscovered errors. Notably, non-agile teams manage to finish a project under high certainty with fewer errors. Consequently, a significant relationship between the development cycle time and the quality performance cannot be identified.

Given the previously defined premises, it can be stated that agile teams perform better than non-agile teams regarding the product quality. Except for one scenario (scenario I, *agile team, long cycles*), all agile teams completed the projects with less than half of the errors that remained undiscovered by non-agile teams. According to these results, team characteristics have more impact on quality performance than the product development method.

To explain why agile teams outperformed non-agile teams in almost all scenarios, it is noteworthy that the configuration of team characteristics for agile teams comprises more team autonomy, so that relatively more errors are produced. For that reason, agile teams must have detected more errors in order to outperform non-agile teams in this way. Agile teams are smaller, their members meet more often, and therefore communicate more intensively with each other. Consequently, agile team members have more awareness about other member's work which all in all leads to a better mutual controlling behavior of the product development system, and thus to better error discovery (Schmidt, 2016).

6. Conclusions and future research

The literature review and empirical data analysis of this paper have shown that agile product development goes beyond the sole application of lightweight project management methods. To successfully develop agility as a capability of involving the customer in the project and quickly

responding to changes, a product development team must frequently interact with the customer, be able to make decisions quickly, and communicate them effectively as identified in (Conforto et al., 2016).
The analysis of the developed SD model showed that frequent development iterations and product releases helped to reduce the project completion time significantly in an uncertain project environment where product changes are likely to occur (Glaiel, 2012). Further, the analysis results suggest that project managers should consider the product development method as the primary project design variable when time is the most important performance objective. Regarding quality performance, agile team characteristics like small team sizes, high team autonomy, and frequent team meetings managed to reduce product errors much more effectively than the variation of iteration and release cycle times (Schmidt, 2016). In case a flawless development of the product is the major priority, the results suggest that project managers should primarily take team characteristics into consideration.
Although this research has addressed some important questions in the field of agility and product development, there is still potential for further research efforts. So far, the model is based on insights from the literature, some empirical data, and existing models. The next logical step is to validate the model with performance data from real projects and examine if it is in line with the experience of practitioners. Thereby, the model can be adjusted to a specific branch or company and tested by conducting case studies. Results from case studies could then also be used to rework the model and improve its validity. Additionally, the developed model has limitations regarding the behavior of some variables that may change dynamically in real projects. For instance, the uncertainty level or the cycle times were kept static throughout the simulation. To create a more realistic model behavior and see if it leads to different insights, these variables need to be redefined and adjusted dynamically.

References

Asan, E. and Bilgen, S. (2013), "Agility Problems in Traditional Systems Engineering – A Case Study", *Proceedings of the Third International Conference on Complex Systems Design & Management CSD&M 2012*, pp. 53-71. https://doi.org/10.1007/978-3-642-34404-6_4

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W. et al. (2001), *Manifesto for Agile Software Development*. [online] agilemanifesto.org. Available at: http://agilemanifesto.org (accessed 30.11.2017).

Cohn, M. (2010), *Succeeding with Agile: Software Development Using Scrum*, Addison-Wesley, Boston. https://doi.org/10.5381/jot.2010.9.4.r1

Conforto, E.C. and Amaral, D.C. (2016), "Agile project management and stage-gate model — A hybrid framework for technology-based companies", *Journal of Engineering and Technology Management*, Vol. 40, pp. 1-14. https://doi.org/10.1016/j.jengtecman.2016.02.003

Conforto, E.C., Amaral, D.C., da Silva, S.L., Di Felippo, A. and Kamikawachi, D.S. (2016), "The agility construct on project management theory", *International Journal of Project Management*, Vol. 34 No. 4, pp. 660-674. https://doi.org/10.1016/j.ijproman.2016.01.007

Conforto, E.C., Rebentisch, E. and Amaral, D.C. (2014a), Project Management Agility Global Survey, Massachusetts Institute of Technology, Consortium for Engineering Program Excellence – CEPE, Cambridge, Massachusetts, U.S.A.

Conforto, E.C., Rebentisch, E. and Amaral, D.C. (2014b), The Building Blocks of Agility as a Team's Competence in Project Management, Massachusetts Institute of Technology, Consortium for Engineering Program Excellence – CEPE, Cambridge, Massachusetts, U.S.A.

Cooper, K.G. (1993), "The rework cycle: benchmarks for the project manager", *Project Management Journal*, Vol. 24 No. 1, pp. 17–21.

Cooper, R.G. (2008), "Perspective: The stage-gate-idea-to-launch-process-update, what's new, and NextGen Systems", *Journal of Product Innovation Management*, Vol. 16 No. 5, pp. 299-310. https://doi.org/10.1111/j.1540-5885.2008.00296.x

Glaiel, F. (2012), Agile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods, Massachusetts Institute of Technology, Engineering Systems Division.

Highsmith, J. (2010), *Agile project management: creating innovative products*, Pearson, Boston.

Hoda, R., Noble, J. and Marshall, J. (2012), "Developing a grounded theory to explain the practices of self-organizing Agile teams", *Empirical Software Engineering*, Vol. 17 No. 6, pp. 609-639. https://doi.org/10.1007/s10664-011-9161-0.

Hull, E., Jackson, K. and Dick, J. (2017), *Requirements Engineering*, Springer, Cham. https://doi.org/10.1007/978-3-319-61073-3_10

Kock, A. and Gemünden, H.G. (2016), "Antecedents to Decision-Making Quality and Agility in Innovation Portfolio Management", *Journal of Product Innovation Management*, Vol. 33 No. 6, pp. 670-687. https://doi.org/10.1111/jpim.12336
Li, W. (2012), Modeling and Managing Engineering Changes in a Complex Product Development Process, PhD thesis, Syracuse University.
MacCormack, A. (2013), *Embracing Uncertainty*. [online] MIT Technology Review. Available at: https://www.technologyreview.com/s/511406/embracing-uncertainty/ (accessed 30.11.2017).
McManus, H. and Hastings, D. (2006), "A Framework for Understanding Uncertainty and its Mitigation and Exploitation in Complex Systems", *IEEE Engineering Management Review*, Vol. 34 No. 3, pp. 81-94. https://doi.org/10.1109/emr.2006.261384
Moe, N.B., Dingsøyr, T. and Dybå, T. (2008), "Understanding Self-organizing Teams in Agile Software Development", *19th Australian Conference on Software Engineering, IEEE, Perth, WA, Australia*, pp. 76-85. https://doi.org/10.1109/ASWEC.2008.28
Petersen, K. and Wohlin, C. (2009), "A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case", *Journal of Systems and Software*, Vol. 82 No. 9, https://doi.org/10.1016/j.jss.2009.03.036
Ramasesh, R.V. and Browning, T.R. (2014), "A conceptual framework for tackling knowable unknown unknowns in project management", *Journal of Operations Management*, Vol. 32 No. 4, pp. 190-204. https://doi.org/10.1016/j.jom.2014.03.003
Royce, W. (1970), "Managing the Development of Large Software Systems", *Proceedings of IEEE WESCON 26, August*, pp. 1-9.
Sage, A.P. and Rouse, W.B. (2009), *Handbook of Systems Engineering and Management*, John Wiley & Sons, New York.
Salas, E., Sims, D.E. and Burke, C.S. (2005), "Is there a 'Big Five' in teamwork?", *Small Group Research*, Vol. 36 No. 5, pp. 559-599. https://doi.org/10.1177/1046496405277134.
Schmidt, C. (2016), *Agile Software Development Teams*, Springer, Heidelberg. https://doi.org/10.1007/978-3-319-26057-0.
Schuh, G. and Bender, D. (2012), "Grundlagen des Innovationsmanagements", In: Schuh, G. (Ed.), *Innovationsmanagement*, Springer, Heidelberg, pp. 1–16. https://doi.org/10.1007/978-3-642-25050-7_1
Schuh, G., Rudolf, S., Riesener, M. and Kantelberg, J. (2016), "Application of Highly-Iterative Product Development in Automotive and Manufacturing Industry", *Charting The Future Of Innovation Management - Proceedings of ISPIM Innovation Forum, March*, pp. 1–13.
Schuh, G., Salmen, M., Jussen, P., Riesener, M., Zeller, V. et al. (2017), "Geschäftsmodell-Innovation", In: Renihart, G. (Ed.), *Handbuch Industrie 4.0 - Geschäftsmodelle, Prozesse, Technik*, Carl Hanser Verlag, München. https://doi.org/10.3139/9783446449893.fm
Schwab, K., Sala-i-Martín, X., Samans, R. and Blanke, J. (2016), The Global Competitiveness Report 2016-2017, World Economic Forum, Geneva, Switzerland.
Serrador, P. and Pinto, J.K. (2015), "Does Agile work? – A quantitative analysis of agile project success", *International Journal of Project Management*, Vol. 33 No. 5, pp. 1040-1051. https://doi.org/10.1016/j.ijproman.2015.01.006
Smith, P.G. (2007), *Flexible Product Development – Building Agility for Changing Markets*, Jossey-Bass, San Francisco.
Sterman, J.D. (2003), *Business Dynamics – Systems Thinking and Modeling for a Complex World*, McGraw-Hill, Boston.
Trott, P. (2005), *Innovation Management and New Product Development*, Pearson, Harlow.
Turner, R. (2007), "Toward Agile Systems Engineering Processes", *The Journal of Defense Software Engineering*, pp. 11-15.
Ulrich, K.T. and Eppinger, S.D. (2015), *Product Design and Development*, McGraw-Hill, New York.
Vinekar, V., Slinkman, C.W. and Nerur, S. (2006), "Can agile and traditional systems development approaches coexist? An ambidextrous view", *Information Systems Management*, Vol. 23 No. 3, pp. 31-42. https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93705.4

Jan Kantelberg, M.Sc.
Laboratory for Machine Tools and Production Engineering (WZL), Innovation Management
Campus-Boulevard 30, 52074 Aachen, Germany
Email: j.kantelberg@wzl.rwth-aachen.de